zenon Treiber Handbuch

MODRTU32

v.7.11

©2014 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

Die Weitergabe und Vervielfältigung dieses Dokuments ist - gleich in welcher Art und Weise - nur mit schriftlicher Genehmigung der Firma COPA-DATA gestattet. Technische Daten dienen nur der Produktbeschreibung und sind keine zugesicherten Eigenschaften im Rechtssinn. Änderungen - auch in technischer Hinsicht - vorbehalten.

Inhalt

1. Willkommen bei der COPA-DATA Hilfe 5

2. MODRTU32 5

3. MODRTU32 - Datenblatt 6

4. Treiber-Historie 7

5. Voraussetzungen 8

5.1 PC 8

6. Konfiguration 8

6.1 Anlegen eines Treibers 9

6.2 Einstellungen im Treiberdialog 10

6.2.1 Allgemein 11

6.2.2 Com 14

6.2.3 Einstellungen 16

6.2.4 Verbindung TCP/IP 20

7. Variablen anlegen 22

7.1 Variablen im Editor anlegen 23

7.2 Adressierung 25

7.3 Treiberobjekte und Datentypen 27

7.3.1 Treiberobjekte 27

7.3.2 Zuordnung der Datentypen 30

7.4 Variablen anlegen durch Import 31

7.4.1 XML Import 31

7.4.2 DBF Import/Export 32

7.5 Treibervariablen 39

8. Treiberspezifische Funktionen 44

9. Treiberkommandos 45

10. Fehleranalyse 47

10.1 Analysetool 47
10.2 Fehlernummern 49
10.3 Checkliste 50

1. Willkommen bei der COPA-DATA Hilfe

ALLGEMEINE HILFE

Falls Sie in diesem Hilfekapitel Informationen vermissen oder Wünsche für Ergänzungen haben, wenden Sie sich bitte per E-Mail an documentation@copadata.com (mailto:documentation@copadata.com).

PROJEKTUNTERSTÜTZUNG

Unterstützung bei Fragen zu konkreten eigenen Projekten erhalten Sie vom Support-Team, das Sie per E-Mail an support@copadata.com (mailto:support@copadata.com) erreichen.

LIZENZEN UND MODULE

Sollten Sie feststellen, dass Sie weitere Module oder Lizenzen benötigen, sind unsere Mitarbeiter unter sales@copadata.com (mailto:sales@copadata.com) gerne für Sie da.

2. MODRTU32

Der Treiber beherrscht das Protokoll "Modbus ASCII" für serielle, "Modbus RTU" für serielle und für TCP/IP Verbindungen bzw. das Protokoll "Open Modbus TCP" für TCP/IP Verbindungen. Der Treiber ist ein Modbus Master. (Für Modbus Slave steht das Process Gateway zur Verfügung).

3. MODRTU32 - Datenblatt

Allgemein:	
Treiberdateiname	MODRTU32.exe
Treiberbezeichnung	Modbus RTU und Open Modbus TCP
Steuerungs-Typen	Alle Steuerungen, die Modbus RTU bzw. Open Modbus TCP oder Modbus ASCII (seriell) unterstützen
Steuerungs-Hersteller	ABB; Gantner; GE Fanuc; Modbus RTU; Mondial; Schiele; Telemecanique; Schneider; Wago; Areva; GE Multilin;

Treiber unterstützt:	
Protokoll	Modbus RTU - serial; Open Modbus TCP; Modbus RTU over TCP; Modbus ASCII - serial;
Adressierung: Adress-basiert	x
Adressierung: Namens-basiert	-
Kommunikation spontan	-
Kommunikation pollend	x
Online Browsing	-
Offline Browsing	-
Echtzeitfähig	-
Blockwrite	x
Modemfähig	-
Serielles Logging	x
RDA numerisch	x

RDA String	x

Voraussetzungen:	
Hardware PC	RS 232 Schnittstelle oder Standard Netzwerkkarte
Software PC	-
Hardware Steuerung	-
Software Steuerung	-
Benötigt v-dll	x

Plattformen:	
Betriebssysteme	Windows CE 6.0, Embedded Compact 7; Windows Vista, 7, 8, 8.1 Server 2008/R2, Server 2012/R2;
CE Plattformen	x86; ARM;

4. Treiber-Historie

Datum	Treiberversion	Änderung
07.07.08	4400	Treiberdokumentation wurde neu erstellt

TREIBERVERSIONIERUNG

Mit zenon 7.10 wurde die Versionierung der Treiber verändert. Ab dieser Version gibt es eine versionsübergreifende Build-Nummer. Das ist die Zahl an der 4. Stelle der Dateiversion.
Zum Beispiel: `7.10.0.4228` bedeutet: Der Treiber ist für Version `7.10`, Servicepack `0` und hat die Build-Nummer `4228`.

Erweiterungen oder Fehlerbehebungen werden zukünftig in einem Build eingebaut und sind dann ab der nächsthöheren Build-Nummer verfügbar.

Beispiel

*Eine Treibererweiterung wurde in Build **4228** implementiert. Der Treiber, den Sie im Einsatz haben, verfügt über die Build-Nummer **8322**. Da die Build-Nummer Ihres Treibers höher ist als die Build-Nummer der Erweiterung, ist die Erweiterung enthalten. Die Versionsnummer des Treiber (die ersten drei Stellen der Dateiversion) spielen dabei keine Rolle. Die Treiber sind versionsunabsabhängig*

5. Voraussetzungen

Dieses Kapitel enthält Informationen zu den Voraussetzungen, die für die Verwendung des Treibers erforderlich sind.

5.1 PC

Um eine serielle Verbindung zur Steuerung herstellen zu können wird am PC ein serieller COM Port Benötigt. Wichtig dabei ist, dass der Port zur Laufzeit nicht von anderer Software (z. B. Modemtreiber, Steuerungssoftware, ...) verwendet und somit blockiert wird.

Dieser Treiber unterstützt eine Verbindung über die Standard Netzwerkkarte des PCs. Achten Sie darauf, dass die Steuerung und der PC im selben Netzwerkbereich liegen und die Subnet-Masken auf beiden Geräten dementsprechend konfiguriert sind.

6. Konfiguration

In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

Info

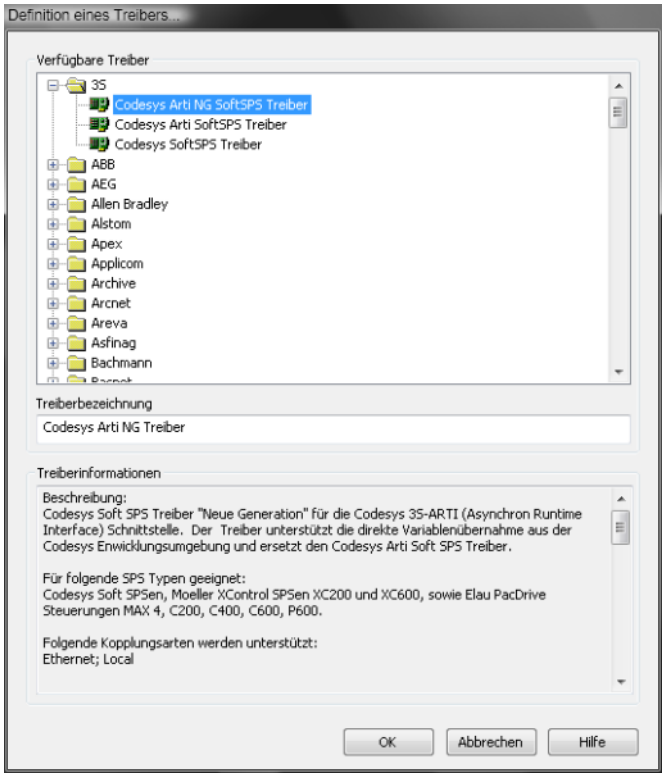
Weitere Einstellungen, die Sie für Variablen in zenon vornehmen können, finden Sie im Kapitel Variablen (main.chm::/15247.htm) der Online-Hilfe.

6.1 Anlegen eines Treibers

Um einen neuen Treiber anzulegen:

1. Klicken Sie mit der rechten Maustaste im Projektmanager auf **`Treiber`** und selektieren Sie im Kontextmenü **`Treiber neu`**.

2. In der folgenden Dialogbox bietet Ihnen das Programm eine Auflistung aller verfügbaren Treiber an.



3. Selektieren Sie den gewünschten Treiber und vergeben Sie eine Bezeichnung für diesen:

- Die Treiberbezeichnung muss eindeutig sein, d.h. wird ein und derselbe Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
- Die Treiberbezeichnung ist Bestandteil des Dateinamens. Daher darf Sie nur Zeichen enthalten, die vom Betriebssystem unterstützt werden. Nicht gültige Zeichen werden durch einen Unterstrich (_) ersetzt.
- **Achtung:** Die Bezeichnung kann später nicht mehr geändert werden.

4. Bestätigen Sie den Dialog mit **OK**. Im folgenden Dialog werden die einzelnen Konfigurationen der jeweiligen Treiber eingestellt.

Für ein Projekt müssen nur die jeweils notwendigen Treiber eingebunden werden. Späteres Einbinden eines weiteren Treibers ist problemlos möglich.

Info

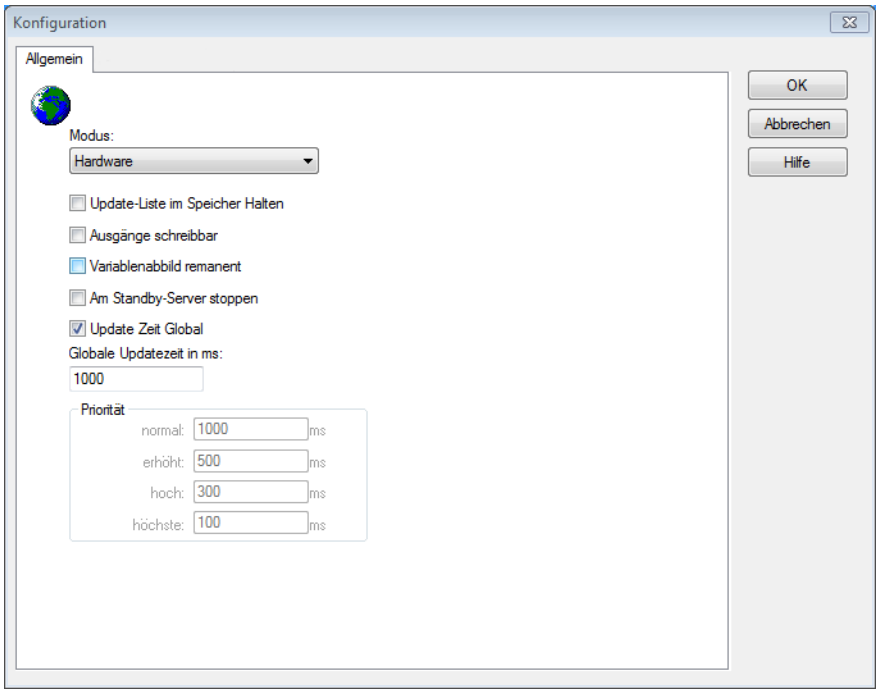
Bei neu angelegten Projekten oder bei bestehenden Projekten, die in eine Version ab 6.21 konvertiert werden, werden die folgenden Treiber automatisch angelegt:

- Intern
- MathDr32
- SysDrv.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein



Parameter	Beschreibung
Modus	Ermöglicht ein Umschalten zwischen Hardware und Simulationsmodus ▸ Hardware: Die Verbindung zur Steuerung wird hergestellt. ▸ Simulation - statisch Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus bleiben die Werte konstant bzw. die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. Zum Beispiel zwei Variablen vom Typ Merker mit Offset 79, können zur Runtime unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der Simulatortreiber. ▸ Simulation - zählend Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. ▸ Simulation - programmiert Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation. (main.chm::/25206.htm)
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	Aktiv: Ausgänge können beschrieben werden. Inaktiv: Das Beschreiben der Ausgänge wird unterbunden.

	Hinweis: Steht nicht für jeden Treiber zur Verfügungen.
`Variablenabbild remanent`	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert wenn: ▸ einer der Status S_MERKER_1(0) bis S_MERKER8(7), REVISION(9), AUS(20) oder ERSATZWERT(27) aktiv ist Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp `Treibervariable` ist ▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation) Folgende Status werden beim Start der Runtime nicht restauriert: ▸ SELECT(8) ▸ WR-ACK(40) ▸ WR-SUC(41) Der Modus **`Simulation - programmiert`** beim Treiberstart ist kein Kriterium, um das remanente Variablenabbild zu restaurieren.
`Am Standby Server stoppen`	Einstellung für Redundanz bei Treibern, die nur eine Kommunikationsverbindung erlauben. Dazu wird der Treiber am Standby Server gestoppt und erst beim Hochstufen wieder gestartet. **Achtung:** Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet. `Aktiv`: Versetzt den Treiber am nicht-prozessführenden Server automatisch in einen Stopp-ähnlichen Zustand. Im Unterschied zum Stoppen über Treiberkommando erhält die Variable nicht den Status **`abgeschaltet`** (**`statusverarbeitung.chm::/24150.htm`**), sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden.
`Update Zeit Global`	`Aktiv`: Die eingestellte `Globale Update Zeit` in `ms` wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht verwendet.

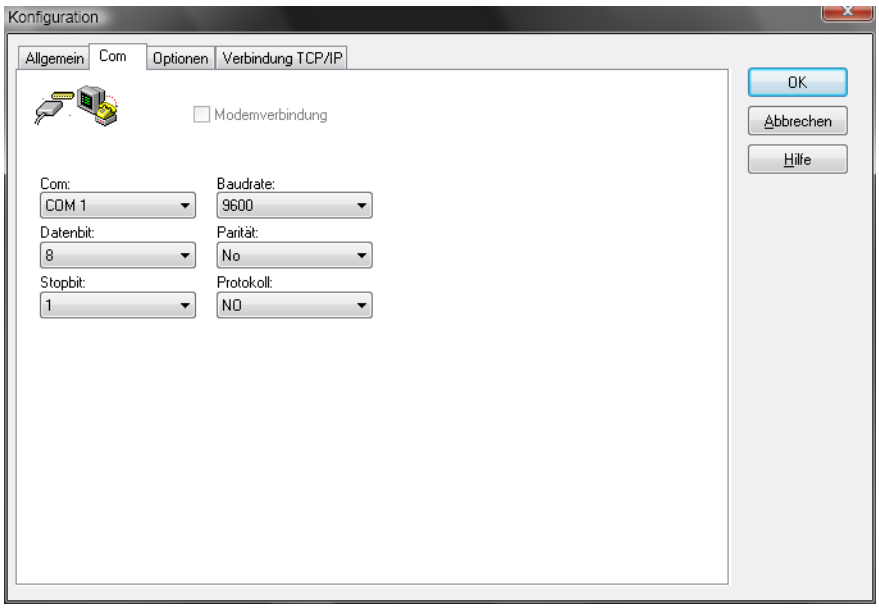
	`Inaktiv`: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.
`Priorität`	Hier werden die Pollingzeiten der einzelnen Prioritäten eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung zu den Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritäten kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit bzw. benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast.
`OK`	Übernimmt Einstellungen aus allen Registerkarten und schließt den Dialog.
`Abbrechen`	Verwirft alle Änderungen und schließt den Dialog.
`Hilfe`	Öffnet die Online-Hilfe.

UPDATE ZEIT ZYKLISCHE TREIBER

Für zyklische Treiber gilt:

Beim **`Sollwert Setzen`**, **`Advisen`** von Variablen und bei **`Requests`** wird sofort ein Lesezyklus für alle Treiber ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. Update-Zeiten können damit für zyklische Treiber kürzer ausfallen als eingestellt.

6.2.2 Com

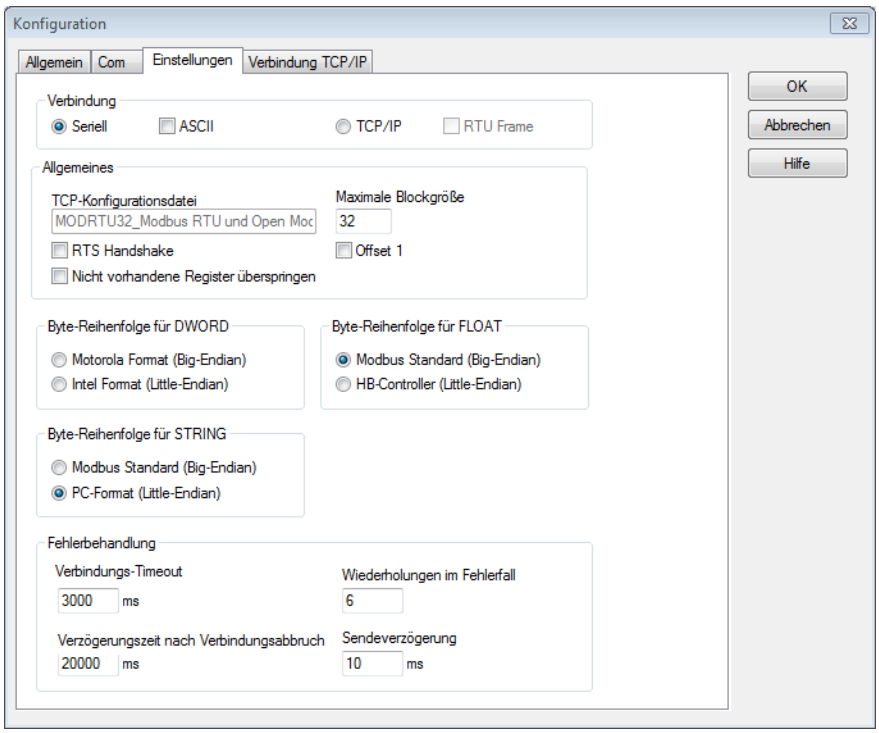


Parameter	Beschreibung
Com	Auswahl serielle Schnittstelle COM1 – COM64
Datenbit	Datenwortlänge in Bit: 5,6,7,8
Stopbit	1, 1.5, 2
Baudrate	Datenübertragungsrate 110; 300; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200
Parität	gerade/ungerade/keine
Protokoll	RTS/CTS/kein

DEFAULT KONFIGURATION

Parameter	Beschreibung
Com	COM1
Baudrate	Anpassen nach Einstellen auf Steuerung
Datenbit	8
Stoppbit	1
Parität	Anpassen nach Einstellung auf Steuerung
Protokoll	NO

6.2.3 Einstellungen



Parameter	Beschreibung
Verbindung	Definition, welche Anbindung verwendet wird.
Seriell	Seriell steht sowohl Modbus RTU als auch Modbus ASCII zur Auswahl
ASCII	Aktiv: Der Treiber kommuniziert mit Modbus ASCII Protokoll. Inaktiv: Der Treiber kommuniziert mit Modbus RTU.
TCP/IP	Benötigt eine Konfigurationsdatei. In dieser werden die TCP/IP Einstellungen gespeichert.
RTU Frame	Aktiv: Serielle Information wird 1:1 ins TCP-Protokoll übertragen.
Allgemeines	Allgemeine Einstellungen.
TCP-Konfigurationsdatei	Hier wird der Name der TCP Konfigurationsdatei festgelegt. Default: **MODRTU32_[Treiberbezeichnung].txt**
Maximale Blockgröße	Maximale Blockgröße eines Datentelegramms in WORDs. **Achtung zenon vor 5.50 SP6**: Blockgröße muss kleiner 64 sein. (1-125, Default 32)
RTS Handshake	Manche Steuerungen benötigen den RTS Handshake zur korrekten Kommunikation.
Offset 1	Der „Offset 1" wirkt sich auf die Adressierung aller Variablen aus. Aktiv: Der Treiber subtrahiert beim Senden die Variablenadressen (Coils, Register) jeweils um 1 und addiert 1 beim Empfang.
Nicht vorhandene Register überspringen	Daten werden vom Treiber in einem Block gelesen. Sind jedoch zwischen zwei nahe liegenden Adressbereichen keine Daten vorhanden, würde der Treiber versuchen, auf diese nicht vorhandenen Bereiche zuzugreifen. Diese Option verhindert dies.
Byte-Reihenfolge für DWORD	Definiert die Reihenfolge von niederwertigem und höherwertigem Wort bei Doppelwort-Objekten (DINT/UDINT). Es kann zwischen Motorola (Big-Endian) und Intel (Little-Endian) gewählt werden.
Motorola Format (Big-Endian)	Aktiv: DWORD Ordering nach Motorola Format.
Intel Format (Little-Endian)	Aktiv: DWORD Ordering nach Intel Format.
Byte-Reihenfolge für FLOAT	Definiert die Reihenfolge von niederwertigem und höherwertigem Wort

	bei FLOAT-Objekten (REAL). Es kann zwischen Modbus Standard (Big-Endian) und HB-Controller (Little-Endian) gewählt werden.
Modbus Standard (Big-Endian)	Aktiv: Floatordering nach Standard Modbus.
HB Controller Float (Little-Endian)	Aktiv: Floatordering nach HB Controller.
Byte-Reihenfolge für STRING	Definiert Darstellung der Byte-Reihenfolge. **Hinweis:** Falls in der Dokumentation einer SPS eine Angabe enthalten ist, gilt meistens MSB-first = Motorola = Big-Endian = Modbus. Und: Intel = Little-Endian = PC. Das kann in manchen Dokumentationen aber unterschiedlich gehandhabt werden.
Modbus Standard (Big-Endian)	Aktiv: Darstellung nach Modbus Standard mit vertauschten Zeichen.
PC-Format (Little-Endian)	Aktiv: Darstellung im PC-Format mit aufsteigender Reihenfolge.
Fehlerbehandlung	
Verbindungs-Timeout	Zeit in Millisekunden, die auf eine Antwort vom Slave gewartet wird. Kommt keine Antwort innerhalb dieser Zeit, wird ein Kommunikationsfehler ausgegeben. Nur bei serieller Verbindung einstellbar. Bei TCP/IP-Verbindung wird für den Verbindungs-Timeout die Einstellung für die Eigenschaft Timeout [s] im Knoten Netzwerk verwendet. **Hinweis:** Die Kommunikations-Timeout Zeit muss auf jeden Fall höher sein als die maximale Antwortzeit des ModbusRTU Slave. Es kann sonst zur Fehlinterpretation der Antworten kommen. Default: 3000 ms
Wiederholungen im Fehlerfall	Anzahl der Sendewiederholungen wenn nach der eingestellten Kommunikations-Timeout Zeit keine Antwort vom Slave erfolgt. 1: ein Verbindungsversuch, keine Wiederholungen 0: ständig Wiederholung Default: 6
Verzögerungszeit nach Verbindungsabbruch	Zeit in Millisekunden die nach dem Auftreten eines Kommunikationsfehlers gewartet wird, bevor ein Wiederaufbau der Verbindung versucht wird.

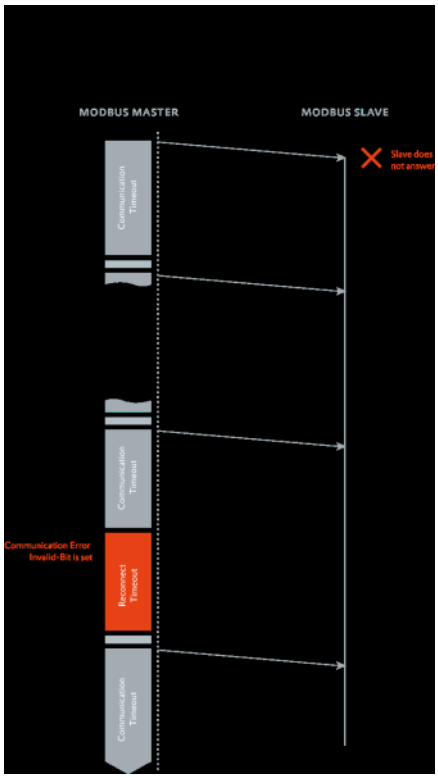
	Default: 20000 ms
Sendeverzögerung	Zeitverzögerung in Millisekunden bei „send"-Aufträgen. Wirkt sich auf den gesamten Treiber aus. Ist nur bei serieller Verbindung auswählbar.
OK	Übernimmt Änderungen und schließt Dialog.
Abbrechen	Verwirft Änderungen und schließt Dialog.
Hilfe	Öffnet Online-Hilfe.

TIMEOUT VERHALTEN

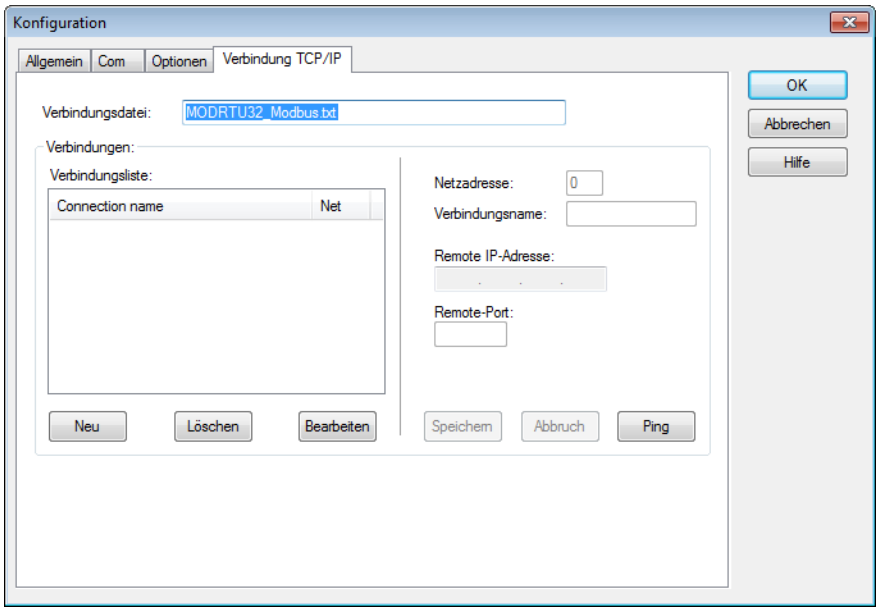
Bei TCP Verbindung ist der Kommunikations-Timeout durch die systemabhängige Timeout [s] vorgegeben.

Nach dem Ablauf der Timeoutzeit versucht der Treiber bei Defaulteinstellungen bis zu 5 Mal, erneut die Anfragen zu senden. Sind diese 5 Versuche nicht erfolgreich, so wird für die Zeit des Reconnect-Timeout kein weiterer Verbindungsaufbau versucht.

Bei serieller Verbindung werden für das Timeout verhalten die Zeiten aus der Treiberkonfiguration verwendet. Vor dem Senden einer Anfrage wird die Sendeverzögerungszeit abgewartet. Nach Senden der Anfrage wird die für den Verbindungs-Timeout einegstellte Zeit auf eine Antwort gewartet. Kommt in dieser Zeit keine Antwort so wird bis zu 5 Mal noch versucht, die Anfrage erneut zu senden. (Default 6 bedeutet 5 zusätzliche Versuche). Danach wird für die eingestellte Reconnect-Timeoutzeit kein weiterer Verbindungsversuch unternommen. Siehe dazu auch folgende Grafik.



6.2.4 Verbindung TCP/IP



Parameter	Beschreibung
Verbindungsdatei	Name der Datei, in der die Einstellungen für die einzelnen TCP/IP-Teilnehmer gespeichert werden. Nur Anzeige. Änderungen des Dateinamens erfolgen in der Registerkarte **Optionen** (auf Seite 16).
Verbindungen	Einstellungen der Verbindungen.
Verbindungsliste	Liste aller definierten Verbindungen zu Steuerungen.
Netzadresse	Jeder Verbindung wird eine Netzadresse zugewiesen. Diese muss mit den Einstellungen in der Eigenschaft Netzadresse bei der Variablendefinition übereinstimmen. Der Wert 0 ist reserviert für Broadcasts. **Hinweis:** Bei MODBUS TCP-Kommunikation über ein Gateway (TCP/IP auf RS485) muss hier die Adresse der Zielstation (die sogenanntes Modbus Unit ID oder SlaveAdresse) verwendet werden. Bei direkter MODBUS TCP-Verbindung wird das **Unit ID** Adressfeld von manchen Geräten ignoriert, da die TCP/IP-Adresse der Steuerung prinzipiell ausreichend ist. In diesem Fall kann hier ein beliebiger Wert eingestellt werden.
Verbindungsname	Name der Verbindung, wie er in der Verbindungsliste angezeigt wird.
Remote IP Adresse	Adresse der Steuerung.
Remote-Port	Port-Adresse der Steuerung. Details finden Sie im Handbuch Ihrer Steuerung. Standardport: 502
Neu	Erstellt eine neue Verbindung.
Löschen	Löscht markierte Verbindung aus Liste.
Bearbeiten	Öffnet markierte Verbindung zum Bearbeiten.
Speichern	Übernimmt alle Änderungen für bearbeitete Verbindung und schließt Editiermöglichkeit.
Abbruch	Verwirft alle Änderungen für bearbeitete Verbindung und schließt Editiermöglichkeit.
Ping	Sendet einen Ping zur IP-Adresse, die für diese Verbindung konfiguriert wurde. Ermöglicht das Testen der Verbindung zum Gerät. Wird der Ping negativ abgeschlossen, prüfen Sie die IP-Adresse und ob das Gerät online ist.
OK	Änderungen im Dialog übernehmen und Dialog schließen.

	Nur verfügbar, wenn keine Verbindung im Zustand "bearbeiten" ist.
Abbrechen	Verwirft alle Änderungen und schließt Dialog.
Hilfe	Öffnet Online-Hilfe.

NEUE VERBINDUNG ANLEGEN

1. klicken Sie auf die Schaltfläche **Neu**
2. Tragen Sie die Verbindungsdetails ein
3. klicken Sie auf **Speichern**

VERBINDUNG BEARBEITEN

1. wählen Sie in der Verbindungsliste die gewünschte Verbindung
2. klicken Sie auf die Schaltfläche **Bearbeiten**
3. ändern Sie die Verbindungsparameter
4. schließen Sie mit **Speichern** ab

VERBINDUNG LÖSCHEN

1. wählen Sie in der Verbindungsliste die gewünschte Verbindung
2. klicken Sie auf die Schaltfläche **Löschen**
3. die Verbindung wird aus der Liste gelöscht

7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

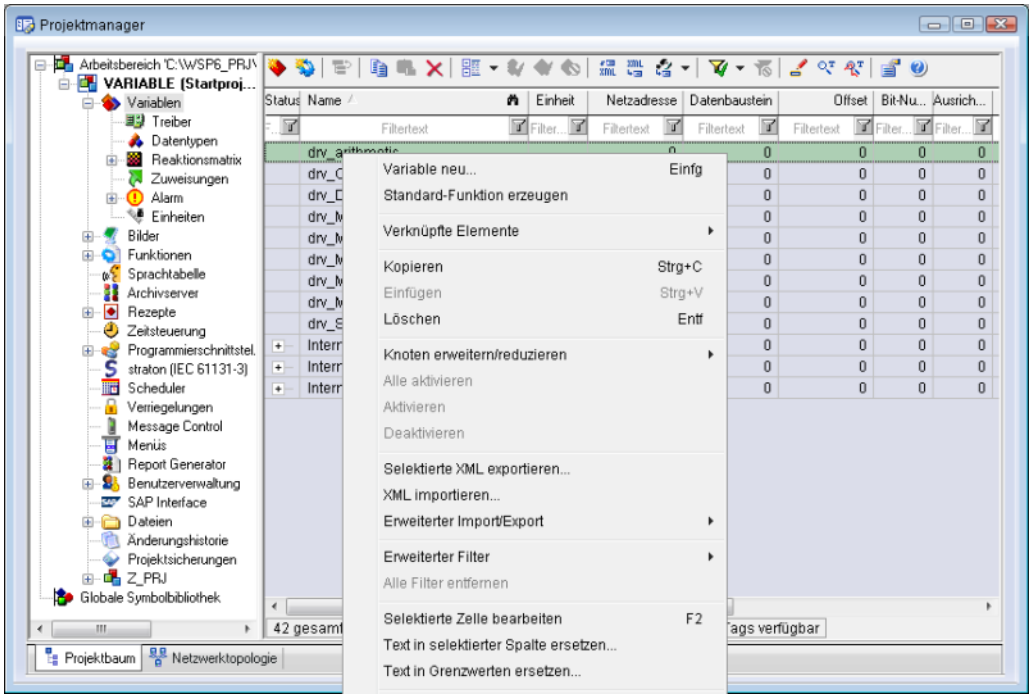
Variablen können angelegt werden:

- als einfache Variable
- in Arrays (main.chm::/15262.htm)
- als Struktur-Variablen (main.chm::/15278.htm)

DIALOG VARIABLE

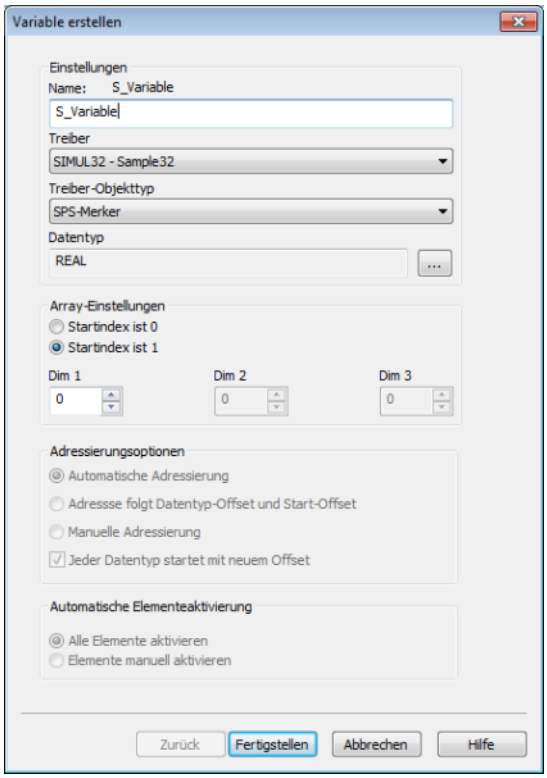
Um eine neue Variable zu erstellen, gleich welchen Typs:

1. wählen Sie im Knoten **`Variablen`** im Kontextmenü den Befehl **`Variable neu`**



2. der Dialog zur Konfiguration der Variable wird geöffnet
3. konfigurieren Sie die Variable

4. welche Einstellungen möglich sind, hängt ab vom Typ der Variablen



Eigenschaft	Beschreibung
Name	Eindeutiger Name der Variablen. Ist eine Variable mit gleichem Namen im Projekt bereits vorhanden, kann keine weitere Variable mit diesem Namen angelegt werden. Maximale Länge: 128 Zeichen **Achtung:** Die Zeichen **#** und **@** sind für Variablennamen nicht erlaubt. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv.
Treiber	Wählen Sie aus der Dropdownliste den gewünschten Treiber. **Hinweis:** Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) geladen.
Treiber-Objekttyp (cti.chm::/28685.htm)	Wählen Sie aus der Dropdownliste den passenden Treiber-Objekttyp aus.

Datentyp	Wählen Sie den gewünschten Datentyp. Klick auf die Schaltfläche ... öffnet den Auswahl-Dialog.
Array-Einstellungen	Erweiterte Einstellungen für Array-Variablen. Details dazu lesen Sie im Abschnitt Arrays.
Adressierungsoptionen	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.
Automatische Elementeaktivierung	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.

ABLEITUNG VOM DATENTYP

Messbereich, Signalbereich und Sollwert Setzen werden immer:

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

Hinweis Signalbereich: Bei einem Wechsel auf einen Datentyp, der den eingestellten Signalbereich nicht unterstützt, wird der Signalbereich automatisch angepasst. Zum Beispiel wird bei einem Wechsel von **INT** auf **SINT** der Signalbereich auf 127 geändert. Die Anpassung erfolgt auch dann, wenn der Signalbereich nicht vom Datentyp abgeleitet wurde. In diesem Fall muss der Messbereich manuell angepasst werden.

7.2 Adressierung

Eigenschaft	Beschreibung
Name	Frei vergebbarer Name, **ACHTUNG:** Je Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung, für Betriebsmittelkennung, Kommentar...
Netzadresse	Busadresse (Slaveadresse) und Netzadresse der Variable. Diese Adresse bezieht sich für serielle Kommunikation auf die Modbusadresse der Steuerung - die Adresse des Empfängers. Bei Kommunikation über Ethernet bezieht sich auch auf die Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welcher Steuerung/Verbindung sich die Variable befindet. Wird über ein Gateway (z.B. TCP/IP auf RS485) mit den einzelnen Modbusstationen kommuniziert, so ist für jede Modbusstation eine eigene TCP/IP-Verbindung anzulegen. Als IP Adresse ist die Adresse des Gateways anzugeben, als Netzadresse der Verbindung und in den Variablen die Adresse der Modbusstation - Modbus Unit ID. Höchste mögliche Modbus Adresse lt. Protokollspezifikation ist 247. Adresse 0 wird für Broadcast messages verwendet (nur schreibend).
Datenbaustein	wird für diesen Treiber nicht verwendet
Offset	Offset der Variable, die Speicheradresse der Variable in der Steuerung. Einstellbar [0.. 4294967295]
Ausrichtung	Der Treiber adressiert Wortbasierend (16Bit), wird nur ein Byte gelesen, so kann hier eingestellt werden ob das HiByte oder das LowByte angesprochen werden soll.
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets. Mögliche Eingabe [0.. 65535] Funktionierender Bereich [0-15]
Stringlänge	Nur verfügbar bei String-Variablen: Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
Treiberobjekttyp	Abhängig vom verwendeten Treiber wird bei der Erstellung der Variable ein Objekttyp ausgewählt und kann hier später geändert werden.
Datentyp	Datentyp der Variable; wird beim Erstellen der Variable ausgewählt und kann hier später verändert werden. ACHTUNG: Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variable überprüft bzw. angepasst werden.

7.3 Treiberobjekte und Datentypen

Treiberobjekte sind in der Steuerung verfügbare Bereiche wie z. B. Merker, Datenbausteine usw. Hier lesen Sie, welche Treiberobjekte vom Treiber zur Verfügung gestellt werden und welche IEC-Datentypen dem jeweiligen Treiberobjekt zugeordnet werden können.

7.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiber-objekttyp	Kanaltyp	Lesen: Modbus Funktion (Code hex/dec)	Schreiben: Modbus Funktion (Code hex/dec)	Unterstützte Datentypen	Kommentar
Alarm Stack	97	0x03/3	N/A	STRING	Spezielle Events aus Secheron Steuerung auslesen. Adressierung: ▸ Event-Typ = Datenbaustein ▸ Subtyp = Offset.
Analog Input	10	0x04/4	N/A	REAL, LREAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING	Class 1 - input register. Lineare Adressierung wie Holding Register.
Byte aligned Coil	69	0x01/1	0x0F/15	BOOL	Class 1 - coil. Lineare Adressierung einstufig über Offset.
Coil	65	0x01/1	0x05/5	BOOL	Class 1 - coil. Lineare Adressierung einstufig über Offset.
Device Status	24	0x11/17	N/A	BOOL, USINT	SPS-spezifisch.
Device Identification	68	0x2B/43	N/A	STRING	SPS-spezifisch.
Input Status	66	0x02/2	N/A	BOOL	Class 1 - input discretes. Lineare Adressierung einstufig über Offset.
Holding Register	8	0x03/3	0x10/16	REAL, LREAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT,	Class 0 - multiple register. Lineare Adressierung: ▸ Bit: einstufig über

				STRING	Offset und Bit-Nummer ▸ Byte (8 Bits): einstufig über Offset und Ausrichtung ▸ Wort (16 Bits) Doppel-Wort (32 Bits) Float (32 Bits) String(n*Byte): einstufig über Offset
Holding Register 32 Bit	70	0x03/3	0x10/16	REAL	Pro Offset werden 32 Bit (4 Byte) gelesen.
Holding Register 64 Bit	71	0x03/3	0x10/16	LREAL	Pro Offset werden 64 Bit (8 Byte) gelesen.
Preset Single Register (FC 6)	67	N/A	0x06/6	UINT	Clase 1 - single register.
Treibervariable	35	J	J	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variablen zur statistischen Auswertung der Kommunikation. **Hinweis:** Übermittelt zwischen Runtime und Treibern, nicht zur SPS. Weitere Infos finden Sie bei den Treibervariablen (auf Seite 39).

MODBUS FUNCTIONCODES

Functioncode hex/dec	Modbus Bezeichner	Kommentar
0x01/1	Read coils	This function code is used to read from 1 to 2000 contiguous status of coils (bits) in a remote device
0x02/2	Read inputs discretes	This function code is used to read from 1 to 2000 contiguous

		status of discrete inputs (bits) in a remote device
0x03/3	Read multiple registers	This function code is used to read a block of contiguous holding registers (1 to 125 words) in a remote device.
0x04/4	Read input registers	This function code is used to read a block of contiguous input registers (1 to 125 words) in a remote device.
0x05/5	Write coil	This function code is used to write a single output (one bit) to either ON or OFF in a remote device.
0x06/6	Write single register	This function code is used to write a single (one word) holding register in a remote device.
0x10/16	Write multiple registers	This function code is used to write a block of contiguous holding registers (1 to approx. 120 words) in a remote device.
0x11/17	Report Slave ID	This function code is used to read the description of the type, the current status, and other information specific to a remote device.
0x2B/43	Read Device Identification	This function code allows reading the identification and additional information relative to the physical and functional description of a remote device.

7.3.2 Zuordnung der Datentypen

Alle Variablen in zenon werden von IEC-Datentypen abgeleitet. In folgender Tabelle werden zur besseren Übersicht die IEC-Datentypen den Datentypen der Steuerung gegenübergestellt.

BEISPIEL FÜR ALLE MÖGLICHEN ZENON DATENTYPEN:

Steuerung	zenon	Datenart
BOOL	BOOL	8
BYTE	SINT	10
BYTE	USINT	9
WORD	INT	1
WORD	UINT	2
DWORD	DINT	3
DWORD	UDINT	4
FLOAT	REAL	5
FLOAT	LREAL	6
STRING	STRING	12

Datenart: Die Eigenschaft `Datenart` ist die interne numerische Bezeichnung des Datentyps. Diese wird auch für den erweiterten DBF Import/Export der Variablen verwendet.

7.4 Variablen anlegen durch Import

Variablen können auch mittels Variablenimport angelegt werden. Für jeden Treiber stehen XML- und DBF-Import zur Verfügung.

> **Info**
>
> *Details zu Import und Export von Variablen finden Sie im Handbuch Import-Export (main.chm::/13028.htm) im Abschnitt Variablen (main.chm::/13045.htm).*

7.4.1 XML Import

Für den Import/Export von Variablen gilt:

- Der Import/Export darf nicht aus dem Globalprojekt gestartet werden.
- Der Start erfolgt über:
  - Kontextmenü zu Variablen bzw. Datentyp im Projektbaum
  - oder Kontextmenü einer Variablen bzw. eines Datentyps
  - oder Symbol in der Symbolleiste Variablen

> **Achtung**
>
> *Beim Import/Überschreiben von existierenden Datentypen werden alle Variablen geändert, die auf diesem existierenden Datentyp basieren.*
>
> ***Beispiel:***
>
> *Es existiert ein Datentyp XYZ abgeleitet vom Typ `INT` mit Variablen, die auf diesem Datentyp basieren. Ihre zu importierende XML-Datei enthält ebenfalls einen Datentyp mit Namen XYZ, allerdings abgeleitet vom Typ `STRING`. Wird dieser Datentyp importiert, so wird der existierende Datentyp überschrieben und bei allen auf ihm basierenden Variablen der Typ angepasst. D.h. die Variablen sind jetzt `STRING`- und keine `INT`-Variablen mehr.*

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

> **Info**
>
> *Import und Export über CSV oder dBase unterstützt keine treiberspezifischen Variableneinstellungen wie z. B. Formeln. Nutzen Sie dafür den Export/Import über XML.*

IMPORT DBF-DATEI

Um den Import zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie in der Dropdownliste von **`Erweiterter Export/Import`** ... den Befehl **`dBase importieren`**

3. folgen Sie dem Importassistenten

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

> **Info**
>
> *Beachten Sie:*
>
> - Treiberobjekttyp und Datentyp müssen in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.
> - dBase unterstützt beim Import keine Strukturen oder Arrays (komplexe Variablen).

EXPORT DBF-DATEI

Um den Export zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie im Dropdownliste von **`Erweiterter Export/Import`** ... den Befehl **`dBase exportieren...`**
3. folgen Sie dem Exportassistenten

> **Achtung**
>
> DBF-Dateien:
>
> - müssen in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
> - dürfen im Pfadnamen keinen Punkt (.) enthalten.
> Z. B. ist der Pfad `C:\users\Max.Mustermann\test.dbf` ungültig.
> Gültig wäre: `C:\users\MaxMustermann\test.dbf`
> - müssen nahe am Stammverzeichnis (Root) abgelegt werden, um die eventuelle Beschränkungen für Dateinamenlänge inklusive Pfad zu erfüllen: maximal 255 Zeichen

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

> **Info**
>
> *dBase unterstützt beim Export keine Strukturen oder Arrays (komplexe Variablen).*

Dateiaufbau der dBase Exportdatei

Für den Variablenimport und -export muss die dBaseIV-Datei folgende Struktur und Inhalte besitzen.

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

- in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
- nahe am Stammverzeichnis (Root) abgelegt werden

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_R	C	128	Ursprünglicher Name einer Variablen, der durch den Eintrag unter VARIABLENNAME ersetzt werden soll (Feld/Spalte muss manuell angelegt werden). Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_D	Log	1	Variable wird bei Eintrag 1 gelöscht (Feld/Spalte muss manuell angelegt werden).
TAGNR	C	128	Kennung. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
EINHEIT	C	11	Technische Maßeinheit
DATENART	C	3	Datentyp (z. B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z. B. Merkerbereich, Datenbereich, ...) entspricht Treiber-Objekttyp.
HWKANAL	Num	3	Bus-Adresse
BAUSTEIN	N	3	Datenbaustein-Adresse (nur bei Variablen aus den Datenbereich der SPS)
ADRESSE	N	5	Offset

BITADR	N	2	Für Bit-Variablen: Bitadresse Für Byte-Variablen: 0=niederwertig, 8=höherwertig Für String-Variablen: Stringlänge (max. 63 Zeichen)
ARRAYSIZE	N	16	Anzahl der Variablen im Array für Index-Variablen ACHTUNG: Nur die erste Variable steht voll zur Verfügung. Alle folgenden sind nur über VBA oder den Rezeptgruppen Manager zugänglich
LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	Float	16	Rohwertsignal minimal (Signalauflösung)
SIGMAX	F	16	Rohwertsignal maximal (Signalauflösung)
ANZMIN	F	16	technischer Wert minimal (Messbereich)
ANZMAX	F	16	technischer Wert maximal (Messbereich)
ANZKOMMA	N	1	Anzahl der Nachkommastellen für die Darstellung der Werte (Messbereich)
UPDATERATE	F	19	Updaterate für Mathematikvariablen (in sec, eine Dezimalstelle möglich) bei allen anderen Variablen nicht verwendet
MEMTIEFE	N	7	Nur aus Kompatibilitätsgründen vorhanden
HDRATE	F	19	HD-Updaterate für hist. Werte (in sec, eine Dezimalstelle möglich)
HDTIEFE	N	7	HD-Eintragtiefe für hist. Werte (Anzahl)
NACHSORT	L	1	HD-Werte als nachsortierte Werte
DRRATE	F	19	Aktualisierung an die Ausgabe (für zenon DDE-Server, in sec, eine Kommastelle möglich)
HYST_PLUS	F	16	Positive Hysterese; ausgehend vom Messbereich
HYST_MINUS	F	16	Negative Hyterese; ausgehend vom Messbereich
PRIOR	N	16	Priorität der Variable

REAMATRIZE	C	32	Name der zugeordnete Reaktionsmatrix
ERSATZWERT	F	16	Ersatzwert; ausgehend vom Messbereich
SOLLMIN	F	16	Sollwertgrenze Minimum; ausgehend vom Messbereich
SOLLMAX	F	16	Sollwertgrenze Maximum; ausgehend vom Messbereich
VOMSTANDBY	L	1	Variable vom Standby Server anfordern; der Wert der Variable wird im redundanten Netzwerkbetrieb nicht vom Server sondern vom Standby Server angefordert
RESOURCE	C	128	Betriebsmittelkennung. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: 0: Nichtlineare Wertanpassung wird verwendet 1: Nichtlineare Wertanpassung wird nicht verwendet
ADJZENON	C	128	Verknüpftes VBA-Makro zum Lesen der Variablenwerte für die nichtlineare Wertanpassung.
ADJWVBA	C	128	Verknüpftes VBA-Makro zum Schreiben der Variablenwerte für die nichtlineare Wertanpassung.
ZWREMA	N	16	Verknüpfte Zählwert-Rema.
MAXGRAD	N	16	Maximaler Gradient für die Zählwert-Rema.

Achtung

Beim Import müssen Treiberobjekttyp und Datentyp in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.

GRENZWERTDEFINITION

Grenzwertdefinition für Grenzwert 1 bis 4, bzw. Zustand 1 bis 4:

Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx 1 steht und hier −1, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen
BTB1	L	1	Protokollierung in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Druckerausgabe (bei CEL oder Alarm)
QUITTIER1	L	1	quittierpflichtig
LOESCHE1	L	1	löschpflichtig
VARIABLE1	L	1	dyn. Grenzwertverknüpfung der Grenzwert wird nicht durch einen absoluten Wert (siehe Feld GRENZWERTx) festgelegt.
FUNC1	L	1	Funktionsverknüpfung
ASK_FUNC1	L	1	Ausführung über die Alarmmeldeliste
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktion (steht hier -1, so wird die bestehende Funktion beim Import nicht überschrieben)
A_GRUPPE1	N	10	Alarm/Ereignis-Gruppe
A_KLASSE1	N	10	Alarm/Ereignis-Klasse
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Farbe als Windowskodierung
GRENZTXT1	C	66	Grenzwerttext
A_DELAY1	N	10	Zeitverzögerung
INVISIBLE1	L	1	Unsichtbar

BEZEICHNUNGEN IN DER SPALTE BEMERKUNG BEZIEHEN SICH AUF DIE IN DEN DIALOGBOXEN ZUR DEFINITION VON VARIABLEN VERWENDETEN BEGRIFFE. BEI UNKLARHEITEN, SIEHE KAPITEL VARIABLENDEFINITION.

7.5 Treibervariablen

Das Treiberkit implementiert eine Reihe von Treibervariablen. Diese sind unterteilt in:

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **`drvvar.dbf`** (auf der CD im Verzeichnis: `CD_Laufwerk:/Predefined/Variables`) verfügbar und können von dort importiert werden.

Hinweis: Variablennamen müssen in zenon einzigartig sein. Soll nach einem Import der Treibervariablen aus **`drvvar.dbf`** ein erneuter Import durchgeführt werden, müssen die zuvor importierten Variablen umbenannt werden.

Info

Nicht jeder Treiber unterstützt alle Treibervariablen.

Zum Beispiel werden:

- Variablen für Modem-Informationen nur von modemfähigen Treibern unterstützt
- Treibervariablen für den Polling-Zyklus nur für rein pollenden Treibern
- verbindungsbezogene Informationen wie ErrorMSG nur von Treibern, die zu einem Zeitpunkt nur eine Verbindung bearbeiten

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	UINT	0	Haupt-Versionsnummer des Treibers.
SubVersion	UINT	1	Sub-Versionsnummer des Treibers.
BuildVersion	UINT	29	Build-Versionsnummer des Treibers.
RTMajor	UINT	49	zenon Hauptversionsnummer
RTMinor	UINT	50	zenon Sub-Versionsnummer
RTSp	UINT	51	zenon Servicepack-Nummer
RTBuild	UINT	52	zenon Buildnummer
LineStateIdle	BOOL	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	BOOL	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	BOOL	24.2	Der Anruf wird angenommen.
LineStateDialtone	BOOL	24.3	Rufton wurde erkannt.
LineStateDialing	BOOL	24.4	Wahl aktiv.
LineStateRingBack	BOOL	24.5	Während Verbindungsaufbau.
LineStateBusy	BOOL	24.6	Zielstation besetzt.
LineStateSpecialInfo	BOOL	24.7	Spezielle Statusinformation empfangen.
LineStateConnected	BOOL	24.8	Verbindung hergestellt.
LineStateProceeding	BOOL	24.9	Wahl ausgeführt.
LineStateOnHold	BOOL	24.10	Verbindung in Halten.
LineStateConferenced	BOOL	24.11	Verbindung im Konferenzmodus.
LineStateOnHoldPendConf	BOOL	24.12	Verbindung in Halten für Konferenz.
LineStateOnHoldPendTransfer	BOOL	24.13	Verbindung in Halten für Transfer.
LineStateDisconnected	BOOL	24.14	Verbindung beendet.
LineStateUnknow	BOOL	24.15	Verbindungszustand nicht bekannt.
ModemStatus	UDINT	24	Aktueller Modemstatus.
TreiberStop	BOOL	28	Treiber gestoppt

			Bei `Treiberstop`, hat die Variable den Wert `TRUE` und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert `FALSE` und kein **OFF**-Bit.
SimulRTState	UDINT	60	Informiert über Status der Runtime bei Treibersimulation.

KONFIGURATION

Name aus Import	Typ	Offset	Erklärung
ReconnectInRead	BOOL	27	Wenn TRUE, dann wird beim Lesen automatisch ein Neuaufbau der Verbindung durchgeführt.
ApplyCom	BOOL	36	Änderungen an den Einstellungen der seriellen Schnittstelle zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyCom zur Folge (aktuell ohne weitere Funktion).
ApplyModem	BOOL	37	Änderungen an den Modemeinstellungen zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyModem zur Folge. Diese schließt die aktuelle Verbindung und öffnet eine neue entsprechend den Einstellungen **PhoneNumberSet** und **ModemHwAdrSet**.
PhoneNumberSet	STRING	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	DINT	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	UDINT	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	BOOL	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	BOOL	5	TRUE, wenn der Treiber in Simulation ist.

TreiberProzab	BOOL	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	BOOL	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	STRING	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	UINT	9	Nummer der seriellen Schnittstelle.
Baudrate	UDINT	10	Baudrate der seriellen Schnittstelle.
Parity	SINT	11	Parität der seriellen Schnittstelle.
ByteSize	USINT	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = 0, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	USINT	13	Anzahl der Stoppbits der seriellen Schnittstelle.
Autoconnect	BOOL	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	STRING	17	Aktuelle Telefonnummer.
ModemHwAdr	DINT	21	Hardwareadresse zur aktuellen Telefonnummer.
RxIdleTime	UINT	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	UDINT	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).
RingCountSet	UDINT	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.

ReCallIdleTime	UINT	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	UINT	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

Name aus Import	Typ	Offset	Erklärung
MaxWriteTime	UDINT	31	Längste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MinWriteTime	UDINT	32	Kürzeste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MaxBlkReadTime	UDINT	40	Längste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
MinBlkReadTime	UDINT	41	Kürzeste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
WriteErrorCount	UDINT	33	Anzahl der Schreibfehler.
ReadSucceedCount	UDINT	35	Anzahl der erfolgreichen Leseversuche.
MaxCycleTime	UDINT	22	Längste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
MinCycleTime	UDINT	23	Kürzeste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
WriteCount	UDINT	26	Anzahl der Schreibversuche.
ReadErrorCount	UDINT	34	Anzahl der fehlerhaften Leseversuche.
MaxUpdateTimeNormal	UDINT	56	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Normal` in Millisekunden (ms).
MaxUpdateTimeHigher	UDINT	57	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Höher` in Millisekunden (ms).
MaxUpdateTimeHigh	UDINT	58	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Hoch` in Millisekunden (ms).

MaxUpdateTimeHighest	UDINT	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höchste** in Millisekunden (ms).
PokeFinish	BOOL	55	Geht für eine Abfrage auf 1, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

Name aus Import	Typ	Offset	Erklärung
ErrorTimeDW	UDINT	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler auftrat.
ErrorTimeS	STRING	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler als String auftrat.
RdErrPrimObj	UDINT	42	Nummer des PrimObjektes, als der letzte Lesefehler verursacht wurde.
RdErrStationsName	STRING	43	Name der Station, als der letzte Lesefehler verursacht wurde.
RdErrBlockCount	UINT	44	Anzahl der zu lesenden Blöcke, als der letzte Lesefehler verursacht wurde.
RdErrHwAdresse	DINT	45	Hardwareadresse, als der letzte Lesefehler verursacht wurde.
RdErrDatablockNo	UDINT	46	Bausteinnummer, als der letzte Lesefehler verursacht wurde.
RdErrMarkerNo	UDINT	47	Merkernummer, als der letzte Lesefehler verursacht wurde.
RdErrSize	UDINT	48	Blockgröße, als der letzte Lesefehler verursacht wurde.
DrvError	USINT	25	Fehlermeldung als Nummer.
DrvErrorMsg	STRING	30	Fehlermeldung als Klartext.
ErrorFile	STRING	15	Name der Fehlerprotokolldatei.

8. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

BLOCKWRITE

Zum effizienteren absetzten von Vielen Sollwerten (z. B. Rezepte) kann man das Blockwrite aktivieren, womit Variablen die im Steuerungsspeicher hintereinander liegen in mit einem Schreibtelegramm beschrieben werden bzw. bei größeren Bereichen das Schreiben in wenige Telegramme zusammengefasst wird anstatt jede Variable einzeln zu beschreiben.

Aktivieren des Blockwrite

Folgender Eintrag muss in die project.ini eingetragen werden:

[MODRTU32]

BLOCKWRITE=1

TRANSACTION IDENTIFICATION

Der Treiber erhöht den Transaction Identifier bei jedem verschickten Paket und erwartet vom Slave eine Antwort mit dem gleichen Transaction Identifier oder `0`.

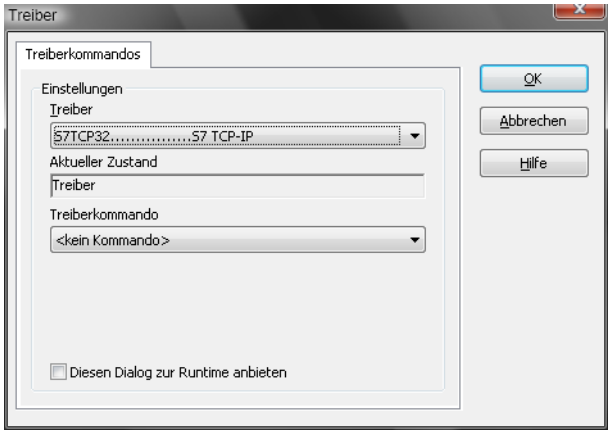
9. Treiberkommandos

Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind. Nicht alle hier beschriebenen Funktionalitäten stehen für jeden Treiber zur Verfügung. Zum Beispiel enthält ein Treiber, der laut Datenblatt keine Modemverbindung unterstützt, auch keine Modem-Funktionalitäten.

Treiberkommandos dienen dazu, Treiber über zenon zu beeinflussen, z. B. starten und stoppen.
Die Projektierung erfolgt über die Funktion **`Treiber Kommandos`**. Dazu:

- legen Sie eine neue Funktion an
- wählen Sie *Variablen -> Treiberkommandos*

- der Dialog zur Konfiguration wird geöffnet



Parameter	Beschreibung
Treiber	Dropdownliste mit allen im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos.
▸ Treiber starten (Online-Modus)	Treiber wird neu initialisiert und gestartet.
▸ Treiber stoppen (Offline-Modus)	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status Abgeschaltet (OFF; Bit 20).
▸ Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ Treiberspezifisches Kommando	Eingabe treiberspezifischer Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
▸ Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist erlaubt.

▸ Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
▸ Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
▸ Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **Treiberkommandos** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzliche Aktionen ausgeführt. Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet, der dann die gewünschte Aktion auf seinem Treiber durchführt. Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

Dadurch ist gewährleistet, dass Server und Standby synchronisiert sind. Dies funktioniert nur, wenn Server und Standby jeweils eine funktionierende und unabhängige Verbindung zur Hardware haben.

10. Fehleranalyse

Sollte es zu Kommunikationsproblemen kommen, bietet dieses Kapitel Hilfen, um den Fehler zu finden.

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame Log-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnose Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter *Start/Alle Programme/zenon/Tools 7.11 -> Diagviewer.*

zenon Treiber protokollieren alle Fehler in Log-Dateien. Der Standardordner für die Log-Dateien ist der Ordner **LOG** unterhalb des Ordners `ProgramData`, zum Beispiel: `C:\ProgramData\zenon\zenon7.11\LOG` für die zenon Version

7.11. Log-Dateien sind Textdateien mit einer speziellen Struktur.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem **`Diagnose Viewer`** kann bei den meisten Treibern die Diagnose-Ebene auf „Debug" und „Deep Debug" erweitert werden. Damit protokolliert der Treiber auch alle anderen wesentlichen Aufgaben und Ereignisse.

Im Diagnose Viewer kann man auch:

- eben erstellte Einträge live mitverfolgen
- die Aufzeichnungseinstellungen anpassen
- den Ordner, in dem die Log-Dateien gespeichert werden, ändern

Hinweise:

1. Unter Windows CE werden aus Ressourcegründen auch Fehler standardmäßig nicht protokolliert.
2. Der Diagnose Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
3. Der Diagnose Viewer zeigt in seiner Standardeinstellung nicht alle Spalten einer Log-Datei an. Um mehr Spalten anzuzeigen, aktivieren Sie die Eigenschaft **`Add all columns with entry`** im Kontextmenü der Spaltentitel.
4. Bei Verwendung von reinem **`Error-Logging`** befindet sich eine Problembeschreibung in der Spalte **`Error text`**. In anderen Diagnose-Ebenen befindet sich diese Beschreibung in der Spalte **`General text`**.
5. Viele Treiber zeichnen bei Kommunikationsprobleme auch Fehlernummern auf, die die SPS ihnen zuweist. Diese werden in **`Error text`** und/oder **`Error code`** und/oder **`Driver error parameter(1`** und **`2)`** angezeigt. Hinweise zur Bedeutung der Fehlercodes erhalten Sie in der Treiberdokumentation und der Protokoll/SPS-Beschreibung.
6. Stellen Sie am Ende Ihrer Tests den Diagnose-Level von **`Debug`** oder **`Deep Debug`** wieder zurück. Bei **`Debug`** und **`Deep Debug`** fallen beim Protokollieren sehr viele Daten an, die auf der Festplatte gespeichert werden und die Leistung Ihres Systems beeinflussen können. Diese werden auch nach dem Schließen des **`Diagnose Viewers`** weiter aufgezeichnet.

Info

Weitere Informationen zum Diagnose Viewer finden Sie im Kapitel Diagnose Viewer (main.chm::/12464.htm).

10.2 Fehlernummern

Folgende Fehlercodes gelten NUR für die Fehlermeldungen

CallSPSread FAILED ! ERROR:' und

'CallSPSwrite FAILED ! ERROR:'

-1	Unzulässige Funktion
-2	Unzulässige MW-Adresse
-3	Unzulässiges Datum
-4	Fehler im Slave
-7	Slave kann die Funktion nicht ausführen
-8	Speicherparitätsfehler im Slave
-10	Timeout bei der Kommunkation mit Slave
-11	Anfrage zum Lesen fehlgeschlagen, da keine Kommunikation zur Steuerung möglich.

Die Felder Para1 und Para2 enthalten zusätzliche Informationen über die Variable die nicht gelesen werden kann:

Die Zahlen beziehen sich auf die interne ID

Para1: Kanaltyp (z.B: 8=Holding Register)

8: Holding Register

10 Input Register

24 Status

65 COIL

66 Input Status

67 Preset Single Register

Para2: Datentyp (z.B: 2= Word, UINT)

1: INT

2: UINT

3: DINT

4: UDINT

5: REAL

8: BOOL

9: SINT

10: USINT

10.3 Checkliste

- Ist der COM Port von einer anderen Anwendung belegt, bzw. stimmen seine Einstellungen?
- Ist das Gerät (die SPS), mit dem versucht wird eine Kommunikation herzustellen, an das Stromversor-gungsnetz angeschlossen?
- Ist das Verbindungskabel zwischen Steuerung (SPS) und PC bzw. IPC korrekt angeschlossen?
- Sind die verwendeten Adressen in der SPS vorhanden?
- Wurde die Logdatei mit dem Diagnose Viewer analysiert? (Welche Fehler sind aufgetreten)?

- Bei Kommunikation über TCP: ist der eingestellte Port frei und wird nicht durch eine Firewall blockiert?
- Test mit Telnet, kann eine Verbindung aufgebaut werden?